DOCUMENT RESUME

ED 247 891 IR 011 240

AUTHOR Every, I. M.
TITLE Graphics and Animation in Teaching Dialogues. CAL Research Group Technical Report No. 14.
PUB DATE Apr 81
NOTE 20p.; Paper presented at the CAL 81 Symposium on Computer Assisted Learning (Leeds, England, April 8-10, 1981). Best available copy.
AVAILABLE FROM Institute of Educational Technology, The Open University, Walton Hall, Milton Keynes, MK7 6AA, England.
PUB TYPE Reports - Descriptive (141) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS Animation; *Computer Assisted Instruction; *Computer Graphics; Computer Simulation; *Courseware; *Design Requirements; Material Development; Physics; *Programing; *Science Instruction; Systems Development
IDENTIFIERS Open University (Great Britain)

ABSTRACT

An Open University project was planned to develop a set of computer assisted learning (CAL) microcomputer programs for a second level, summer school physics course, scheduled to begin in 1982. Major project aims were the development of an underlying system to effectively use medium resolution graphics with a simple animation capability; integration of simulations with tutorial style dialogs--in more or less natural English--which are not necessarily related directly to the simulation itself; and use of a flexible choice of "escape" options. This paper reviews the major issues which were involved in the design and implementation of the system as a whole and looks briefly at some of the individual teaching programs developed, which were planned to include about eight large teaching packages in kinematics, motion in force fields, electromagnetic fields, quantum mechanics, and statics. Concluding remarks indicate that UCSD PASCAL was a good language for the system development and writing of the teaching dialogs and simulations, that memory-mapped graphics devices have a tremendous potential in CAL, and that the most efficient method of animation seems to be via a direct data transfer to the screen memory. Four references are listed. (LMM)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED247891

THE OPEN UNIVERSITY

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)
This document has been reproduced as received from the person or organization originating it.
Minor changes have been made to improve reproduction quality.
Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

Computer Assisted Learning Research Group

Graphics and Animation in Teaching Dialogues

I. M. Every

CAL Research Group Technical Report No. 14

IR011240

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
E.S. Henderson
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

Walton Hall,
Milton Keynes,
England MK7 6AA

Graphics and Animation in Teaching Dialogues

I. M. Every
Academic Computing Service
The Open University
Walton Hall
MILTON KEYNES MK7 6AA
U.K.

This paper was presented at the CAL-81 Conference held at the University of Leeds in April 1981.

IR011240

GRAPHICS and ANIMATION in TEACHING DIALOGUES

I. M. Every, Student Computing Service,
THE OPEN UNIVERSITY

Short Abstract:

Work is currently being carried out at the Open University to develop CAL packages for a new second level physics course (S271 - Discovering Physics). This course will be making extensive use of CAL as part of its summer school programme. The project utilises microcomputers with medium-resolution memory-mapped graphics. Exciting developments in the use of graphics and simple animation have proved to be possible and it is anticipated that these will make a significant contribution to the educational value of the programs. A major development aim has been to achieve the integration of simulations with tutorial-style dialogues in more or less natural English. Such dialogues are not necessarily related directly to the simulation itself and the implementation of a flexible choice of "escape" options has a high priority. This paper reviews the major issues which were involved in the design and implementation of the system as a whole and looks briefly at some of the individual teaching programs developed.

GRAPHICS and ANIMATION in TEACHING DIALOGUES

I. M. Every, Student Computing Service,

THE OPEN UNIVERSITY

INTRODUCTION

This paper arises out of preliminary work on a project being carried out at the Open University by myself in conjunction with members of the Physics Department. The primary purpose of the project is to develop a set of CAL programs which will be used as an integral part of the summer school for a second-level physics course entitled "S271 - Discovering Physics". This course goes into full operation in January 1982 with approximately 650 students and 6 summer schools, each of one week duration. In 1981 the course is being developmentally tested with about 30 students and one summer school week at Durham University.

As part of the development of the teaching programs, one of the main tasks has been to develop an underlying system for making effective use of medium resolution graphics with a capability for simple animation. The ways in which students can interact with a CAL package are also of great importance and a high priority has been given to the development of tools which will allow the student to issue commands at any time and give responses in natural English.

Graphics is particularly important for CAL in the science area where diagrams and graphs are a basic and essential means of communication. Moreover, the additional capability of moving or changing images in real time opens up a tremendous range of possibilities. It enables a program to show visually the actual development of a system with time and develop a student's understanding of the relationship of this with the mathematical and graphical abstractions which are used to describe and "explain" it.

GRAPHICS SYSTEMS FOR CAL

An important requirement for animation in real time is that it should be possible to make major modifications to an image (including partial erasures) very rapidly. Communication via a serial interface (even a high speed one) is not likely to be adequate unless the graphics device has the capability of performing some animation itself. This sort of device tends to be beyond the price range of most CAL. One form of graphics which is becoming extremely common with microcomputer systems is the so-called "memory-mapped" display where the bit pattern in an area of the microcomputer's main memory is displayed on a television set or monitor. This can be extremely flexible and relatively inexpensive. It was decided to use this type of display for the project in hand.

From our experience I would suggest that the following features are desirable in a memory-mapped display:

(i) The pixels (the points corresponding to each bit in memory) should be arranged on a square grid

(ii) The size of the pixel "dots" should be significantly smaller than their separation. This allows the eye to perform some rounding of its own.

(iii) The memory locations corresponding to the screen should be readily and speedily accessible from the high level language. In some systems, such as the Apple II, the memory areas are not contiguous (because of problems with synchronising with the television trace).

(iv) The block of memory which is being displayed as the screen should be selectable in some way.

For most CAL, we believe that it is best to have just a single screen for both text and dialogue. Text is required on the graphics screen for labels, etc. and a separate screen or teletype for textual dialogue can destroy the smoothness of the interaction. Nevertheless, it should be possible to modify text and graphics independently over the whole screen area. One common approach is to dedicate a portion of the screen (say 4 lines) to scrolling text and to use the remainder for graphics. This is probably not adequate for other than simple interactions. The Apple microcomputer has the capability

of allowing characters to be displayed (in reverse mode) on the graphics screen. If this is not further overlaid with graphics this can allow the text to be deleted but the system has limitations. An extended character set (usually of 192 characters) is extremely useful, particularly if this can be easily modified. For physics CAL, a character set which includes mathematical symbols such as integral signs as well as superscripts, subscripts and greek letters seems satisfactory.

In this project it was decided to adopt the strategy of developing both the system software and CAL teaching programs on a relatively powerful machine and then using this software with a number of cheaper machines purchased at a later date. This makes compatibility an important issue. UCSD Pascal was chosen for the project because it is a powerful and user-friendly operating system which is available on a wide range of microcomputers. As a language it has many advantages which encourage good programming practice and make it convenient for the implementation of the underlying CAL system software.

The microcomputer which is being used for development is a TERAK 8510/a with additional 8512 disc drive. This machine is based on a DEC LSI-11 (16 bit) processor with 56K bytes of memory. These machines are widely used with UCSD Pascal and a large amount of graphics and utility software is provided. It is a fast, rugged machine with an extremely flexible medium-resolution graphics system (320 x 240 pixels).

It has been used for CAL development by several groups, including the group at UCI headed by Alfred Bork [1] and the group at the University of Utah headed by Richard C. Brandt [2].

The "delivery" machines have not as yet been selected. Possibilities include:

(i) Apple (with pascal card and other enhancements)

(ii) Vector Graphics (with graphics board)

(iii) Research Machines 380Z (with its HR graphics board)

(iv) North Star Horizon (with an S100 graphics board)

The delivery machine will probably be used in a single disc configuration. A small core of low-level routines will be needed on the delivery machine. The group at UCI have used this technique very successfully to enable their CAL/graphics software to run on both the TERAK and the Apple II machines.

GRAPHICS AND ANIMATION TECHNIQUES FOR CAL

The traditional way of building up a computer-graphics image is by drawing lines. With a memory mapped display, this is not necessarily the most efficient method either in terms of the plotting time or the space required for the data. Nevertheless, for line drawings involving straight or gently curving lines this approach is obviously highly suitable. Software will be required for "drawing" lines and this will usually be implemented in machine code. When efficiency is important (see the later discussion) such software should not

attempt any bounds checking or clipping.

For very small graphical objects consisting of a very small number of pixels the most efficient coding is probably a list of the relative co-ordinates of the individual pixels. The object can then be displayed (with any offset required) by setting the appropriate elements of the boolean array which constitutes the screen memory.

For complex line drawings the so-called "chain-encoding" methods can be very efficient [3], particularly in the amount of data that must be stored. These methods basically record the direction of each pixel relative to the "previous" one in an economical way. They rely for their efficiency on an assumption that the direction of a line is usually fairly constant at the pixel level.

For graphical objects containing large solid areas, one method is to construct the boundary lines and use an algorithm to "fill" the specified areas. This method is efficient as regards the amount of data to be stored but it involves a large number of memory accesses and can be very slow. Another method is called "run-length encoding" [3], where solid areas are represented by series of horizontal (or vertical) lines of pixels. Only the x co-ordinates of the ends of each line need to be stored.

With a medium resolution graphics display quite large graphical objects occupy reasonably tractable areas of memory. Thus it can be extremely fast to display an image by a direct

data transfer from an array to the screen memory. For more precise positioning and drawing in several "modes" a procedure such as the standard UCSD Pascal procedure DRAWBLOCK is required. The "reverse" mode is particularly useful since drawing an object twice in this mode (over the top of any existing image) returns the screen to its original state. This "direct transfer" method can be used either to display graphical data which has been already created by one of the methods outlined above, or it can use data directly from disc. The latter system requires that the data will always be played back on a machine with the same aspect ratio of pixel separations. To create disc files containing this data a graphics editor such as GREDIT supplied for the TERAK is required. This can be an extremely quick and easy method of creating and playing back graphics. For simplicity, we have found it convenient to restrict ourselves to using "blocks" of data each containing a 64x64 pixel array or 16 smaller sub-divisions of 16x16 pixels.

With any graphics display, one has to allow for the possibility that a graphical object may lie at least partly outside the screen area. For a memory mapped display this can lead to a disastrous corruption of memory. A simple but hopelessly inefficient strategy would be to check each pixel just before it is "plotted". Efficiency dictates that the bounds checking should be done at as high a level as possible. For example, it is only necessary to check that a line's end-points are within the screen area or, when animating in a straight line, it is only necessary to check the bounds of the

extreme images. Also, if one is plotting a sequence of contiguous lines it is wasteful to check the common end-points more than once. It seems that the greatest flexibility and efficiency are obtained by separating the checking from the "drawing". This principle is applied in the graphics procedures implemented by the project. The disadvantage of checking at a high level is that specialised "clipping" algorithms must be applied.

The simplest form of animation can be achieved using a storage-tube display with a point tracing out a line. To actually give the appearance of a moving point it is necessary to remove the old points. This is easy to do on a memory-mapped display if one can access the screen memory directly. When animating a more complex graphical object against a background, the main difficulties arise from overlaps between the object in consecutive positions and between the object and the background. In general it is necessary to restore the background (either by drawing a saved copy of the local background area or plotting the object again in reverse mode) and then drawing the object at the new position. If the two positions overlap this can lead to a very distracting flashing effect. To avoid this, it may be necessary to calculate the differences between the images and just plot these [4]. This can be exceedingly simple and efficient if the motion is uniform but otherwise it may be difficult to achieve. For objects consisting of a small number of straight lines animation is possible by drawing and redrawing in reverse mode at a sequence of positions. This can sometimes give rise to a strange "wriggling" effect unless the screen is

blanked or alternated. The simplest method which allows fast and smooth animation seems to be to directly transfer blocks of data to the screen memory. The data can either be obtained from disc or created using one of the methods described earlier in the section.

THE DIALOGUE SYSTEM

The first requirement of a screen-based CAL system is the provision of facilities for controlling the placement of displayed text and student input. The system as implemented by this project is a simplified version of the textport system developed at UCI. It is a two "port" system with one port at the top of the screen dedicated mainly to command dialogues and help messages,and the rest of the screen (with dynamically settable margins) for other text and input. In practice, this has proved to be very simple yet extremely flexible. The system constantly keeps track of the co-ordinates of the last character displayed (or input) so as to allow "interrupts" to dialogues at the top of the screen and resumption where the original dialogue left off.

All student input is done at the lowest level by a Pascal procedure which gets a single character (without echoing it) and checks to see if it is one of the characters used for "escaping" from the current dialogue (to another dialogue) or for requesting help. The procedures for inputting numbers,

commands, letters and response strings all use this low-level procedure and in addition allow the student to make corrections.

If the student presses the HELP key then a suitable message is printed at the top of the screen and the cursor returns to the end of the current input. The help message disappears as soon as the student continues with the input. If the student presses the ESCAPE key then a command line will be printed at the top of the screen. In addition to extra commands provided by the teaching program, commands are automatically provided which will allow the student to ask for acedemic help - to continue,or to leave the program.

One of the most satisfying and satisfactory ways of interacting with a CAL program is via a dialogue in natural English. Until efficient semantics-extracting routines are available probably the most convenient way of decoding "student responses" is to match classes of anticipated strings. This technique is used in author languages such as STAF and PILOT. Recently, a number of enhancements have been proposed for the STAF matching routines. These combine the word based matching of STAF with the wild-card based matching of PILOT. The matching system we have developed is based on this specification and has proved to be very powerful. It is implemented as a boolean function with a single string argument.

Some CAL authors who use interpreted author languages have suggested that PASCAL is not suited to writing complex dialogues. However, with the appropriate tools (such as a small number of system procedures for matching and setting a

number of system flags) PASCAL can be highly suited to programming dialogues. In addition, it has the flexibility which only a full programming language can offer. Interfacing with simulations is much easier than with an interpreted author language.

THE TEACHING PROGRAMS

The "Discovering Physics" course will be using the microcomputers in four ways:-

(i) Simulations and tutorial dialogues will be incorporated in two of the three $1\frac{1}{2}$-day experiment-groups. During the course of the lab. sessions the 4-5 microcomputers will be in constant use and it is hoped that each student will spend several hours alone or in a small group with the programs.

(ii) It is intended that the microcomputers will be used as part of a conventional physical experiment for the analysis of data.

(iii) Tutorial sessions will take place mainly in the evenings and some of these may well be based on one of the computer simulations.

(iv) It is hoped that the microcomputers can be made freely available during the evenings and some of the more entertaining programs could find considerable use.

It is proposed that about eight large teaching packages will be made available in the following areas:

(i) Kinematics

The program will establish links between observed motions (some of which will be controlled by the student) and the graphs and quantities used to describe them.

(ii) Motion in Force Fields

Using simple animation this package should improve the students' understanding of Newton's Laws, momentum conservation, Keplers Laws, orbits and projectile motion. Several "game" situations are being incorporated including a matching orbits problem and an open-ended project involving multi-centre forces.

(iii) Electromagnetic Fields

This program is designed to improve the students' knowledge of the relationship between a field and the underlying charge or current distribution. Included in this package will be a "game" whereby one student (or the computer) can specify a charge or current distribution and a second student will have to deduce this from sections of the potential or field distribution.

(iv) Quantum Mechanics

The main aim is to demonstrate some of the properties of Schroedinger's equation. By modifying the energy parameter students should be able to deduce the eigenvalues for a potential (perhaps chosen by another student).

(v) Statics

This will be a tutorial (with diagrams) on the laws of mechanical equilibrium.

(vi) Other packages on statistical mechanics, special relativity and electromagnetic induction are being planned.

CONCLUDING REMARKS

The project is still a long way from completion, but it is becoming evident that the choice of UCSD Pascal was a good one, both for the development of the system and for the writing of the teaching dialogues and simulations. It is also becoming clear that memory-mapped graphics devices have a tremendous potential in CAL. The most efficient method of animation seems to be via a direct data transfer to the screen memory. If transfers of graphics between machines with different aspect ratios of pixel separation are NOT contemplated, then graphics can conveniently be created using a graphics editor to produce the appropriate disc files. Otherwise the block of data to be transferred to the screen is probably best produced by drawing

lines in a device independent way.

It is anticipated that the system being developed will be widely applicable to simulations and tutorial dialogues. It is hoped to use the system for other Open University courses with summer schools.

REFERENCES

1. Bork A. Computers and Education, 4, pp37-57 (1980)

2. Wood K.W. et al Computers and Education, 4, pp59-66 (1980)

3. Baudelaire P. and Stone M., Computer Graphics (SIGGRAPH-80 procs), 14(3), pp314-319 (1980)

4. Newman W.M. and Sproull R.F., Principles of Interactive Computer Graphics (2nd Edition) McGraw-Hill, New York (1979)

List of Technical Reports

These reports may be obtained (free of charge but subject to availability) from:-

Ann Jones, Institute of Educational Technology, The Open University, Walton Hall, MILTON KEYNES, MK7 6AA, England.

Report No.	Title and Author
1	M.A. Brazer (1980) Using Computers in Distance Education: The first ten years of the British Open University.
3	M.A. Brazer (1980) COMAL-80 - Adding Structure to BASIC.
4	A. Jones (1980) An evaluation of the SOLO programming language: A pilot study of its use at summer school.
5	A. Jones, T. O'Shea (1981) An evaluation of tutorial CAL at the Open University: the use of Cicero in SM286.
6	E.L. Whitelegg (1981) An evaluation of tutorial CAL at the Open University: the use of MERLIN in S291.
7	M.W. Lewis (1981) Improving SOLO's User-Interface: An Empirical study of user behaviour and a proposal for cost-effective enhancements to SOLO.
8	A.T. Vincent (1980) Some applications of a microcomputer.
9	T. O'Shea, A. Floyd (1981) Recording Children's mathematical behaviour.
10	A.T. Vincent (1981) Computer-assisted support for blind students: the use of a microcomputer-linked voice synthesiser.
11	P. Bacsich (1981) Viewdata-style delivery mechanisms for CAL.
12	P.J. Murphy (1981) Biological simulations in distance learning.
13	P.J. Murphy (1981) Computer simulation in Education: Genetics and Evolution.
14	I. Every (1981) Graphics and animation in teaching dialogues.
15	P.J. Murphy, E. Scanlon, E.L. Whitelegg (1981) An evaluation of simulation CAL in the Science Faculty during 1980.
16	M. Eisenstadt, J. Laubsch, J.H. Kahney (1981) Creating pleasant programming environments for cognitive science students.
17	J. Laubsch, M. Eisenstadt (1981) Domain specific debugging aids for novice programmers.
18	J. Roberts (1981) Are fuzzy grids really necessary?
19	J. Preece (1981) Investigating how students interpret cartesian graphs.
20	E. Scanlon, Tom Vincent (1981) CAL in Science: Some Regional Implications.
21	A. Jones(1981) The representation of conceptual models: some issues.
22	E. Scanlon (1982) Improving Problem Solving in Physics.
23	M. Sharples (1982) CYCLOPS - A case study in the design of user-friendly educational technology.
24	T. Hausmann (1982) Mac [illegible].
25	A. Jones (1981) How do novices learn programming?
26	A.G. Priest (1981) A design for an Intelligent Mechanics Tutor.
27	A. Jones, and E. Scanlon; (Eds) A review of research in the CAL group: A report of the first annual conference, November 1981.
28.	R. Everts (1982) A production systems account of children's errors in fraction subtraction